1M

УДК 61:004.45

В.П. Марценюк, А.В. Семенець
Тернопільський державний медичний університет ім. І.Я. Горбачевського, Україна
marceniuk@yahoo.com, semteacher@mail.ru

Підсистема підготовки завдань інформаційної системи перевірки знань у медичній освіті

Метою даної роботи є практична реалізація алгоритму автоматичного проектування тест-білетів. Розглянуто вимоги до інструментальних засобів і зроблено вибір інструментального середовища розробки й системи керування базою даних. Показано лістинги процедур генерації списку питань для тест-білета і рандомізації списку відповідей на питання.

Вступ

Проблема якісного контролю знань особливо актуальна для медичної освіти. Одним з інструментів для одержання керівної педагогічної інформації є результати тестування. Результат стандартизованого тестування дозволяє зіставити рівень окремого об'єкта (абітурієнта, групи) з предмета в цілому (або ж із окремих тем) із середнім рівнем або з подібним об'єктом.

Однак використання тестових завдань – набору питань із варіантами відповідей – замість повноцінних тестів, що нерідко має місце, приводить до спрощення тестування, до простого опитування й дискредитує сам метод тестування, хоча в закордонній практиці він визнаний одним з найбільш надійних засобів масового контролю досягнень абітурієнта [1]. У 1968 році Ф. Лорд і М. Новик [2], [3] сформулювали основні постулати математичної моделі класичної теорії тестування.

У зв'язку зі зростаючим використанням сучасної комп'ютерної техніки при визначенні рівня підготовки абітурієнтів і її широким впровадженням у практику роботи освітніх організацій виникає завдання формалізації процедур і методів підготовки тестових завдань, створення технології тестування, розрахованої на масового користувача.

Під час розробки нового програмного забезпечення (ПО) варто враховувати основні тенденції останніх років у галузі засобів розробки додатків (Integrated Development Environment, IDE).

Перша – це суттєве скорочення комерційної складової ринку засобів розробки – інструменти із самостійних продуктів перетворюються в засоби підтримки тих або інших платформ (Windows, Linux). Як наслідок, провідними постачальниками інструментів стали компанії, для яких основний бізнес – це платформне ПО (такі, як IBM, Microsoft, Sun).

Друга тенденція – поділ платформних технологій на два табори: Microsoft .NET й Java/open source. У першому випадку мова йде фактично про монопродукт Visual Studio, а в другому є досить широкий спектр конкуруючих інструментів, у тому числі й один з одним. Останнім часом також спостерігається процес об'єднання засобів Java-розробки навколо проектів Eclipse й NetBeans. Однак така інтеграція йде на користь насамперед постачальникам платформного ПО.

Третій важливий момент: власне процес розробки ПО починає розглядатися як один з етапів керування життєвим циклом додатків (Application Life Managment – ALM). Саме тому, якщо раніше поняття «інструмент» майже повністю асоціювалося

з інтегрованим середовищем розробки IDE, що включає редактор коду, компілятор, компонувальник й відлагоджувач, то тепер стали частіше говорити про інструментальну систему, до складу якої входять засоби підтримки групової роботи, керування вимогами, тестування і т.д. [4].

Незважаючи на зазначені тенденції, можливість створювати додатки для різних платформ із використанням єдиного середовища розробки залишається дуже актуальною. Серед багатьох фірм, що займаються створенням засобів розробки, компанія Borland (www.borland.com) – практично єдиний великий «неплатформний» постачальник. Флагманський продукт підрозділу IDE компанії Borland – CodeGear – RAD Studio являє собою сукупність Delphi, C++ Builder й Delphi for NET, що використовують мови програмування Object Pascal й C++. Основні переваги, які Delphi, C++ Builder й RAD Studio дають розроблювачам ПО, – це універсальність і швидкість розробки, а також принцип відкритості.

Відмінним прикладом можливостей компонентно-орієнтованого середовища розробки служить продукт CodeGear Delphi/400, що складається з Delphi 2007 for Win32 і набору компонентів і ПО проміжного рівня ClientObjects. Завдяки поєднанню можливостей середовища розробки для Windows зі спеціальним ПО стало можливо створювати й поширювати сучасні GUI-додатки для платформи System I (IBM). Найближчі плани розвитку Delphi й C++ Builder містять у собі випуск Unicode-версії цих продуктів (2008 р.) і 64-розрядної версії (2009 р.) [5].

Мета даної роботи – представити одну практичну реалізацію алгоритму автоматичного проектування тест-білетів.

1. Опис експериментального проекту ІСПЗМО

Сучасні інформаційні системи (ІС), у тому числі й освітні, являють собою конкретні програмні додатки, кожен з яких працює з певним набором інформації – базою даних [6]. Під час розробки нової інформаційної системи насамперед слід зазначити важливість інтеграції із уже існуючими ІС, які використовуються або плануються до використання в даному навчальному закладі. Без інтеграції й спільного використання даних всіх ІС неможлива побудова єдиного інформаційного простору сучасного навчального закладу.

У Тернопільському державному медичному університеті ім. І.Я. Горбачевського (ТДМУ) зараз проводиться експеримент із впровадження комплексного тестування як одного з методів контролю знань студентів. Для технічного забезпечення підготовки й проведення тестування на кафедрі медичної інформатики розробляється **інтегроване середовище перевірки знань у медичній освіті (ІСПЗМО)**. Дана система відноситься до інформаційно-керуючих систем (ІКС) [7].

Основні функції розроблюваної ІКС:
– створення, редагування й зберігання структури навчальних дисциплін;
– створення, редагування й зберігання списку тестових завдань до навчальних дисциплін;
– створення, редагування й зберігання списку студентів навчального закладу;
– створення й зберігання тестових завдань;
– формування завдань для проведення тестування (в он-лайновому або офф-лайновому режимі);
– одержання й зберігання результатів тестування;
– формування звітності за результатами тестування;
– обмін даними з АСУ «Контингент».

Загальний вигляд моделі інформаційної системи перевірки знань в медичній освіті показаний на рис. 1.

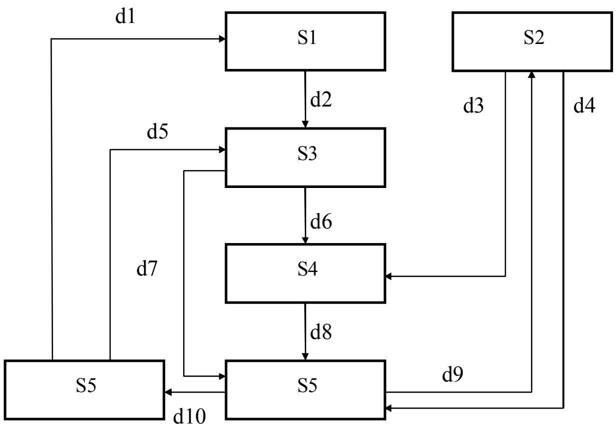


Рисунок 1 – Модель інформаційної системи перевірки знань в медичній освіті

На рис. 1 зображено: 1) структурні блоки моделі: S1 – підсистема формування бази тестових завдань; S2 – підсистема руху контингенту студентів; S3 – підсистема формування тест-білетів; S4 – підсистема реалізації (проведення) іспиту; S5 – підсистема оцінювання результатів тестування; S6 – підсистема аналізу валідності тестів; 2) потоки даних моделі: d1 – оцінка якості тестових питань; d2 – база тестових завдань; d3, d4 – база даних студентів; d5 – оцінка якості тестових білетів; d6 – база тестових білетів; d7 – еталонні відповіді; d8 – список відповідей студентів; d9 – база даних оцінок студентів; d10 – база даних результатів тестування.

2. Практична реалізація алгоритму автоматичного проектування тест-білетів

Розглянемо деякі загальні технічні питання, які виникають при необхідності автоматично спроектувати тест-білет:

1. Необхідна повна реалізація структури тесту – повинні бути підготовлені питання з кожного розділу тест-білета.
2. Кількість розділів і питань з кожного розділу повинна відповідати вказаним.
3. Вибірка питань повинна відбуватися випадковим чином. На етапі пробного тестування частота використання питань у тест-білетах повинна бути рівномірною.
4. Структура білета повинна бути збережена для наступного аналізу й корекції.

Як система керування базою даних для технічної реалізації проекту ІСПЗМО була обрана СУБД Firebird [8]. Основні причини такого вибору:

– при високій продуктивності й надійності, Firebird – вільно розповсюджуване, безкоштовне програмне забезпечення, що досить актуальне для навчального закладу;
– дана СУБД використовується в АСУ «Контингент», що призначена для автоматизації управлінської діяльності в медичному навчальному закладі;
– СУБД має дуже хорошу підтримку в сучасних інструментальних системах для різних мов програмування, таких як Object Pascal, Java , PHP, C++ та інших.

Спрощена блок-схема розробленого авторами алгоритму автоматичного проектування тест-білетів була описана раніше [9]. На даному етапі розробки ІСПЗМО даний алгоритм реалізований мовою Object Pascal з використанням інструментального середовища Delphi. Текст процедури автоматичної генерації тест-білета наведений у Лістингу 1:

Лістинг 1. Процедура генерації тест-білетів

```
procedure GenerateAsksList;
var
  minFreq, AskBySciens, CurrAsk, tmpminpos : integer;
  minFreqCount, MinFreqAskId, NowGenerated , LeaveGenerate, CycleLimit : integer;
  generated : boolean;
  minFreqStr, CurrSciensID : string;
  CurrRightAnswerOrder : TStrings;
  AskIDList, AnswCountList : array of integer;
  AskIDStatus : array of boolean;
begin
   CurrRightAnswerOrder := TStringList.Create;
   randomize;
   testeditdm.TestModDataSet.first;
   While not testeditdm.TestModDataSet.Eof do
   begin          {1}
     CurrSciensID := testeditdm.TestModDataSet.FieldByName('CONTENTMODULE_ID').AsString;
     AskBySciens := testeditdm.TestModDataSet.FieldByName('ASKCOUNT').AsInteger;
     PerformSearchMinimalFrequency(CurrSciensID, minFreqStr);
     NowGenerated := 0;
     while NowGenerated < AskBySciens do
     begin         {2}
       LeaveGenerate := AskBySciens - NowGenerated;
       PerformSearchListOfAskwithMinFreq (CurrSciensID, minFreqStr, minFreqCount);
       setlength(AskIDList, minFreqCount);
       setlength(AnswCountList, minFreqCount);
       setlength(AskIDStatus, minFreqCount);
       GetAskIDList(AskIDList, AnswCountList, AskIDStatus, minFreqCount);
       if minFreqCount > 0 then
       begin       {2+}
         if minFreqCount >= LeaveGenerate then CycleLimit := LeaveGenerate
         else CycleLimit := minFreqCount;
         for CurrAsk := 1 to CycleLimit do
         begin         {3}
           generated := false;
           repeat
             tmpminpos := random(minFreqCount);
             if AskIDStatus[tmpminpos] = false then generated := true;
           until generated = true;
           MinFreqAskId := AskIDList[tmpminpos];
           AskIDStatus[tmpminpos] := true;
           SetAnswerOrder(AnswCountList[tmpminpos], CurrRightAnswerOrder);
           WriteOfflineAskData(MinFreqAskId, CurrRightAnswerOrder, CurrSciensID);
```

```
43             WriteChangedFrequent(MinFreqAskId, minFreqStr);
44             inc(NowGenerated);
45           end           {/3}
46         end         {/2+}
47         else
48         begin       {2++}
49           minFreq := strtoint(minFreqStr)+1;
50           minFreqStr := inttostr(minFreq);
51         end;        {/2++}
52       end;          {/2}
53       testeditdm.TestModDataSet.Next;
54     end;            {/1}
55     CurrRightAnswerOrder.Free;
56   end;
```

Отже, розглянемо роботу даної процедури. Під час запуску відбувається ініціалізація тимчасового списку правильних відповідей CurrRightAnswerOrder, ініціалізація генератора випадкових чисел і перехід у початок списку змістових модулів тестової дисципліни (рядки 11 – 13 Лістингу 1).

У рядках 14 – 17 оголошується зовнішній цикл (цикл 1) процедури, у якому перевіряється, чи досягнуто кінець списку змістових модулів для тестової дисципліни. З таблиці **TEST_MODULES** бази даних зчитуються значення коду змістового модуля CurrSciensID і кількості питань AskBySciens, які повинні бути включені в тестове завдання.

Процедура PerformSearchMinimalFrequency (рядок 18) здійснює визначення мінімальної частоти повторень питань minFreqStr для даного змістового модуля. При цьому фактично виконується наступний SQL-запит:

```
SELECT min(testasks.frequent) min_frequent
FROM testasks
WHERE ((testasks.sciens_id=CurrSciensID)AND(archive=0));
```

На кожній ітерації циклу 1 кількість вибраних питань установлюється в 0 (рядок 19). Наступний за рівнем вкладення цикл 2 (рядок 20) служить для визначення кількості вже відібраних для тестового завдання, зі значеннєвого модуля, питань. При ініціалізації цього циклу визначається кількість питань, які залишилося вибрати LeaveGenerate (рядок 22).

Процедура PerformSearchListOfAskwithMinFreq (рядок 23) визначає в базі даних кількість питань із зазначеною частотою повторів minFreqCount. При цьому виконується наступний SQL-запит:

```
SELECT testasks.frequent, testasks.answcount, testasks.id_ask
FROM testasks
WHERE ((testasks.sciens_id=CurrSciensID)
AND(testasks.frequent=minFreq)AND(archive=0));
```

Далі (рядки 24 – 26) довжина трьох динамічних масивів задається рівною min FreqCount. Динамічні масиви було вирішено використати з метою підвищення швидкодії процедури, тому що ІСПЗМО призначена для роботи, у першу чергу, у локальній комп'ютерній мережі навчального закладу. Оскільки швидкодія ОЗУ комп'ютера в десятки разів більше, ніж мереж стандарту ETHERNET, виграш часу досить значний. Призначення цих масивів наступне:

– AskIDList – містить список кодів питань із мінімальною частотою повторів;

– AnswCountList – містить кількість відповідей для кожного питання;
– AskIDStatus – містить прапор використання питання в даному тестовому завданні.

Наступна процедура GetAskIDList (рядок 27) отримує дані з бази даних і поміщає їх у динамічні масиви, створені на попередньому кроці. Дана процедура повторно виконує SQL-запит, аналогічний запиту процедури PerformSearchListOfAskwithMin Freq, а далі, у циклі, записує отримані дані в динамічні масиви. Слід зазначити, що тут же уточнюється значення кількості питань із заданою частотою повторів. Необхідність у цьому виникає внаслідок можливості паралельної роботи декількох модулів підготовки тестових завдань із однією й тією ж фізичною базою даних. Як результат – дані можуть змінюватися в рамках іншої транзакції. Наслідком зазначеної особливості програми є необхідність у додатковій перевірці кількості питань із мінімальною частотою повторів minFreqCount (рядок 28). Якщо кількість таких питань дорівнює нулю, мінімальна частота повторів збільшується на одиницю й цикл 2 повторюється (рядок 49).

Наступний за рівнем вкладення цикл 3 (рядок 32) призначений для формування списку питань зазначеної кількості у випадковому порядку. Попередньо відбувається визначення кількості повторів циклу CycleLimit, що може бути рівним кількості питань, які залишилося вибрати LeaveGenerate або кількості питань із мінімальною частотою повторів minFreqCount (рядок 30). При кожній ітерації цього циклу відбувається вибірка одного питання випадковим способом. Власне, це відбувається усередині останнього циклу – на четвертому рівні вкладення (рядки 35 – 38) – шляхом визначення статусу питання на деякій позиції в динамічному архіві AskIDStatus. При виявленні в масиві AskIDStatus питання зі статусом «використаний не був», код питання отримується з відповідної позиції масиву AskIDList і зберігається в змінній MinFreqAskId, а сам статус змінюється на протилежний (рядки 39 – 40).

Процедура SetAnswerOrder (рядок 41) призначена для рандомізації списку відповідей на обране питання. Створений список порядку проходження відповідей зберігається разом з варіантом тестового білета. Текст даної процедури наведений у лістингу 2:

Лістинг 2. Процедура рандомізації списку відповідей

```
procedure SetAnswerOrder(tmpAnswCount:integer; var tmpRightAnswOrder:TStrings);
var
  tmporder : set of byte;
  tmpanswpos, i : integer;
begin
  tmporder := [];
  tmpRightAnswOrder.Clear;
  for i := 1 to tmpAnswCount do
  begin
    repeat
      tmpanswpos := random(tmpAnswCount)+1;
    until not (tmpanswpos in tmporder);
    include(tmporder,tmpanswpos);
    tmpRightAnswOrder.Append(inttostr(tmpanswpos));
  end;
end;
```

Дві наступні процедури вносять зміни безпосередньо в базу даних (рядки 42 – 43):
– WriteOfflineAskData – додає обране питання до вибраних у попередніх ітераціях циклів процедури;

– WriteChangedFrequent – зберігає зміни частоти повторень для обраного питання.

Далі – збільшення кількості згенерованих питань NowGenerated і цикл 3 повторюється (рядок 44). Після генерації зазначеної кількості питань відбувається перехід до наступного змістового модуля для тестової дисципліни (рядок 53), і цикл 2 з усіма вкладеннями повторюється.

Висновки

У даній роботі практично реалізований алгоритм автоматичного проектування тест-білетів. Розглянуто вимоги до інструментальних засобів і зроблено вибір інструментального середовища розробки та системи керування базою даних. Показано тексти процедур генерації списку питань тест-білета і рандомізації списку відповідей на питання.

Метою подальших досліджень повинна стати розробка процедури оптимізації тест-білета з урахуванням результатів аналізу валідності тестових питань.

Література

1. Булыгин В.Г. Основы автоматизации процесса обучения / Булыгин В.Г. – Йошкар-Ола, 2003. – 190 с.
2. Lord P.M., Novic M.R. Statistical Teories of Mental Test Scores. Reading, Mass.: Addison-Wesley, 1968.
3. Novic M.R. The Axioms and Principle Results of Classical Test Theory / M.R. Novic // Journal of Mathematical Psychology. – 1966. – № 3. – P. 1-18.
4. Что происходит на рынке средств разработки // BYTE/Россия. – 2008. – № 7-8. – С. 20-21.
5. Ковязин А. Средства разработки CodeGear / А. Ковязин // BYTE/Россия. – 2008. – № 7-8. – С. 23-26.
6. Урнов В. Базы данных – основа всего / В. Урнов // ИКТ в образовании. – 2007. – № 4. – С. 5-6.
7. Энциклопедия кибернетики: в 2 т. – К. : Гол.ред. УРЕ, 1974.
8. http://sourceforge.net/projects/firebird
9. Семенець А.В. Аспекти реалізації інтегрованого середовища перевірки знань в медичній освіті / А.В. Семенець // Медична інформатика та інженерія. – 2008. – № 2. – С. 78-82.

В.П. Марценюк, А.В. Семенец

Подсистема подготовки заданий информационной системы проверки знаний в медицинском образовании

Целью данной работы является практическая реализация алгоритма автоматического проектирования тест-билетов. Рассмотрены требования к инструментальным средствам и произведен выбор инструментальной среды разработки и системы управления базой данных. Показаны тексты процедур генерации списка вопросов для тест-билета и рандомизации списка ответов на вопросы.

V.P. Martsenyuk, A.V. Semenets

The Subsystems Job Processing of the Test Knowledge Information System in the Medical Education

The purpose of this work is practical implementation of the algorithm of automatic test sets construction. Application development tools requirements are considered and integreated development environment and the database management system are selected. The automatic test sets construction and the answers list randomization procedures listings are shown.

Стаття надійшла до редакції 04.03.2009.